FACULTÉ DES SCIENCES ET TECHNIQUES
DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Ingénierie Logicielle et Intégration
des Systèmes Informatiques

TAILLIS LABS
Software artistry at its finest

Mémoire de projet de fin d'étude

Sujet :

Conception et développement d'une solution assurance santé digitale

PEGA HEALTH

Membres du jury:

- Pr. **A. BOULMAKOUL**
- Pr. **A. EL BOUZIRI**
- Pr. **M. SADDOUNE**

Encadrants

- Pr. **A. BOULMAKOUL** Encadrant Pédagogique
- Mr. **Y. FARHOUN** Encadrant Taillis Labs

Réalisé par :

HATIM Ihssane

Année universitaire : 2020/2021

Dédicaces

A mes chers parents,

A mes chers frère et sœur,

A ma grande famille,

A mes professeurs,

A mes encadrants,

A l'équipe Taillis Labs,

Les mots semblent parfois si dénués de sens qu'il est difficile de trouver l'expression qui traduisent mon respect, et ma reconnaissance pour tous les efforts que vous consentez à mon égard. Ce travail n'aurait pu prendre forme sans votre soutien inconditionnel. Puisse Dieu vous accorder sa sainte miséricorde, santé et longue vie.

Hatim Ihssane

Remerciements

On remercie Dieu le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce projet.

Je tiens à remercier vivement mon encadrant le **professeur BOULMAKOUL AZEDINE** ainsi que tous les personnels de Taillis Labs et plus particulièrement **Mr. HARRAZ ABDELGHAFOUR**, **Mr. FARHOUN YOUSSEF** et **Mr. AJAANIT TAOUFIK**, pour leurs énormes efforts, aussi leurs encouragements et leurs glorieux sacrifices, et pour leur suivi dans la durée de stage. Je ne peux pas oublier de préciser toute la profonde affection à mes parents, ma sœur, mon frère et mes proches pour leur soutien durant toute la période de stage.

Mes remerciements vont aussi aux membres du jury, qui m'ont fait l'honneur d'accepter de juger mon travail.

Hatim Ihssane

Résumé

Afin de simplifier le système de santé, l'équipe Taillis Labs a pris l'initiative de concevoir, développer et rendre accessible à toute personne une application mobile/web, permettant à ses divers utilisateurs de bien gérer leurs pièces médicales allant des visites médicales jusqu'aux demandes de remboursement.
Pour alimenter les données au sein de cette application et gérer la logique métier, nous avons mis en place des APIs suivant une architecture micro service assurant un haut niveau d'extensibilité et maintenabilité.
Tout cela en mode agile, pour une communication efficace au sein de l'équipe, une fréquence stable de livraison et une réponse rapide aux changements.

Abstract

In order to introduce new features to the healthcare systems, Taillis Labs team has anticipated to design, develop and make accessible to everyone a mobile/web application, allowing its various users to properly manage their medical records, from medical checkup to reimbursement.
To manipulate all the application's data and manage the business logic, we have designed and developed various APIs following a micro service architecture ensuring a high level of scalability and maintainability.
All in an agile mode, for fast and smooth communication, stability of software delivery and flexibility to any upcoming change.

Table des matières

Glossaire

ADMIN Administrateur de l'application. 4, 6

BR Base de remboursement. 22

DN Denormalizer. 12

DNS Système de Noms de Domaine. 14

RH Responsable des ressources humaines. 4, 6

WN Worker Node. 13

Table des figures

Liste des tableaux

Introduction

Pendant longtemps, les seuls canaux de communication entre un assureur (ou une mutuelle) et son assuré étaient le courrier (exemple de lettre de réclamation) et le téléphone. La relation client se limitait essentiellement à la gestion administrative côté mutuelle, et le suivi de ses règlements côté assuré. Chaque demande, ou modification de contrat santé était plutôt longue et fastidieuse.

Le secteur de l'assurance connaît aujourd'hui une montée en puissance de la pression concurrentielle due à la multiplication des comparateurs d'assurance et l'émergence de nouveaux concurrents ayant des nouvelles approches de fidélisation de clients.

En parallèle, le secteur de l'assurance doit s'adapter au nouveau mode de vie de ses clients qui ont parfaitement intégré le digital dans leurs modes de consommation au quotidien. Cela s'illustre par la volonté de bénéficier d'une meilleure expérience client grâce à la personnalisation des offres et l'omnicanalité.

Il est donc nécessaire pour les assureurs d'optimiser à la fois la qualité de leurs services mais également l'expérience client, qui devient un véritable moteur de croissance pour les acteurs économiques.

Pour répondre à ces besoins, Taillis Labs propose une solution globale de services autour de l'assurance santé. Cette solution innovante aide à avoir la bonne information, la bonne prévention, et le soin adapté tout en gérant rapidement les remboursements.

Pega Health promet de simplifier la vie des indépendants en leur proposant une assurance santé « 0 papier » et « 100 % transparente ». En bref, offrir un service moderne tout en allégeant les formalités administratives liées à la santé.
Ce travail sera décrit dans ce rapport, dont le plan est structuré comme suit :

- Contexte général du projet
- Etude préliminaire (Sprint 0)
- Sprints :
    * Sprint 1 : Gestion de profils ;
    * Sprint 2 : Gestion de plans d'assurance ;
    * Sprint 3 : Gestion des remboursements.

Chapitre 1

Contexte général du projet

Introduction :

Dans ce chapitre, nous allons présenter l'organisme d'accueil Taillis Labs. Ensuite, nous nous focaliserons sur la présentation du contexte du projet afin de décrire le cadre et l'objectif de l'application Pega Health. Enfin, nous verrons la solution retenue qui sera détaillée plus loin dans ce rapport.

1.1 Organisme d'accueil

TAILLIS LABS
Software artistry at its finest

Taillis Labs [1] est une entreprise marocaine de conseil en transformation numérique et de création de solutions digitales qui répondent parfaitement aux problématiques et aux besoins de clients, installée à Casablanca, elle a été crée en 2015 et fondée par Mr. Abdelghafour HARRAZ. Taillis Labs [1] se constitue des principaux départements suivants :

- Direction ;
- Département RH ;
- Département technique ;
- Département marketing et communication ;
- Département commercial.

Parmi les valeurs de Taillis Labs [1] : qualité, agilité, respect, excellence, transparence. L'intelligence artificielle, développement logiciel et design thinking sont au cœur du métier bien que l'innovation au sein des équipes.
Taillis Labs [1] intervient dans plusieurs secteurs d'activités : Automobile, santé, banque, assurance et bien d'autres.
Les clients qui font confiance à Taillis Labs [1] : ALPHA ASSURANCES , LA MAROCAINE VIE, BANK OF AFRICA, AUTOPLUS, ATW, BMCI...

1.2 Contexte général de projet

L'un des secteurs les plus touchés aujourd'hui par la transformation digitale dans le monde est le secteur des assurances.
Les compagnies d'assurance doivent faire face aux exigences de leurs clients, à leurs besoins d'instantanéité et de réactivité. Pour avoir un impact significatif de la digitalisation de systèmes d'assurance au Maroc et pour qu'une telle application puisse satisfaire les besoins des clients et des assureurs, cette application doit englober d'un côté , des fonctionnalités qui assurent à la fois une expérience fluide avec une interface facile à utiliser par l'utilisateur et garder la confidentialité des données des clients, ainsi que l'intégration des données entre les hôpitaux et les compagnies d'assurance.

D'autre côté, l'application doit être capable d'aider les compagnies d'assurance dans la gestion de leurs processus d'affaires et des flux de données. En adoptant la digitalisation, les compagnies d'assurance peuvent suivre l'évolution rapide des besoins des clients dans ce monde numérique.

Le slogan de Taillis Labs [1] « software artistry at its finest » : chez Taillis Labs [1], la qualité du code et des technologies utilisées dans le développement des applications sont aussi importantes et affectent la qualité globale du logiciel.

Le choix des technologies de développement doit satisfaire plusieurs critères pour atteindre les objectifs dans des parfaits délais et avec les résultats attendus. L'aspect principal d'un logiciel fiable est la qualité du code. Une mauvaise qualité de code engendrerait des défaillances lors de maintenance ou au cas de scalabilité horizontale .

1.3 Enjeux et objectifs

L'objectif de ce projet est de réaliser une plateforme web/mobile qui permettra d'organiser et d'automatiser la gestion des processus assurances santé. Pega Health est un projet mené par la société Taillis Labs [1]. Il s'agit d'une offre digitale globale de services autour de l'assurance santé. A travers cette application nous souhaitons avoir :

- Pour l'adhérent : un espace personnel via l'application, à partir duquel il peut gérer l'ensemble de ses démarches santé : ajouter un dossier médical, demander un remboursement ou encore trouver un médecin à proximité ... ;
- Pour un RH : l'application intègre et met à disposition une solution complète qui regroupe l'ensemble des services facilitant la gestion des soins des salariés ;
- Pour le Médecin/Pharmacien : la plateforme permet aux médecins et pharmaciens d'avoir une vue globale sur les dossiers des patients stockant les antécédents médicaux, les médicaments, les bilans d'analyse et les plans de traitement dans une base de données centralisée ;
- Pour l'ADMIN : la plateforme offre une console d'administration pour contrôler tous les aspects de l'application ;
- Pour les organisations : la plateforme offre une gestion des plans d'assurance propre à chaque assureur.

1.4 Conclusion :

Ce chapitre nous a permis de détailler le cadre général du système. En présentant l'entreprise d'accueil, la plateforme que nous allons concevoir et les objectifs visés. Dans ce qui suit, nous allons entamer la première phase de la conception de notre projet, l'étude préliminaire, pour identifier les différentes fonctionnalités du système.

Chapitre 2

Etude préliminaire

Introduction :

L'étude préalable est une étape essentielle pour comprendre le contexte du projet, définir les acteurs de notre système et fixer la méthodologie de gestion pour organiser les tâches et atteindre les objectifs.
L'objectif de cette section est de décrire l'environnement interne et externe de l'application Pega Health à travers une étude des besoins non fonctionnels, fonctionnels et techniques. Nous allons modéliser ces besoins par des diagrammes de cas d'utilisation, d'acteurs, et de contexte statique, les besoins techniques seront modélisés par des diagrammes de déploiement.

2.1 Spécification et modélisation des besoins

2.1.1 Besoins non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes techniques, ergonomiques et esthétiques auxquelles est soumis le système pour sa réalisation et pour son bon fonctionnement. Le système doit vérifier les métriques suivantes :

— fiabilité : la capacité de notre système à rendre les résultats correctes sans échec même en étant affecté par la panne d'un composant (logiciel ou matériel) ;
— La sécurité : l'application doit sécuriser les informations de ses utilisateurs et contrôler l'accès aux ressources ;
— Optimiser et écrire un code clair pour permettre des futures évolutions ou améliorations ;
— Bien documenter l'application.

2.1.2 Besoins fonctionnels

La future application doit satisfaire les besoins suivants :

— La console d'administration ;
— Une e-carte de couverture qui rassemble les informations de santé les plus pertinentes de nos membres, où nous retrouvons les résultats de laboratoire, les ordonnances et les antécédents toujours accessibles à travers des liens partageables tout en contrôlant leurs droits d'accès ;
— La gestion de remboursement permettant aux patients de suivre en temps réel l'état de leurs demandes de remboursement ;
— Un espace RH/admin facilitant les démarches de souscription aux plans de couverture aux personnels des entreprises clientes.

2.1.3 Modélisation des besoins fonctionnels

L'objectif de la modélisation des besoins fonctionnels est de déployer une première version de cahier de charges illustrant l'usage général de l'application et les attentes de chaque utilisateur.

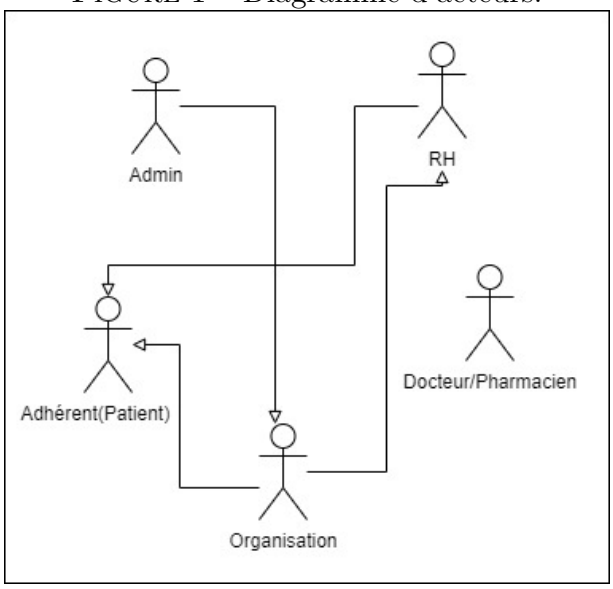
2.1.3.1 Diagramme d'acteurs

Les acteurs de l'application Pega Health seront :

— Adhérent (patient)
— ADMIN ;
— RH ;

— Organisation : peut être une compagnie d'assurance ou une entreprise qui possède des salariés ;
— Médecin/Pharmacien.

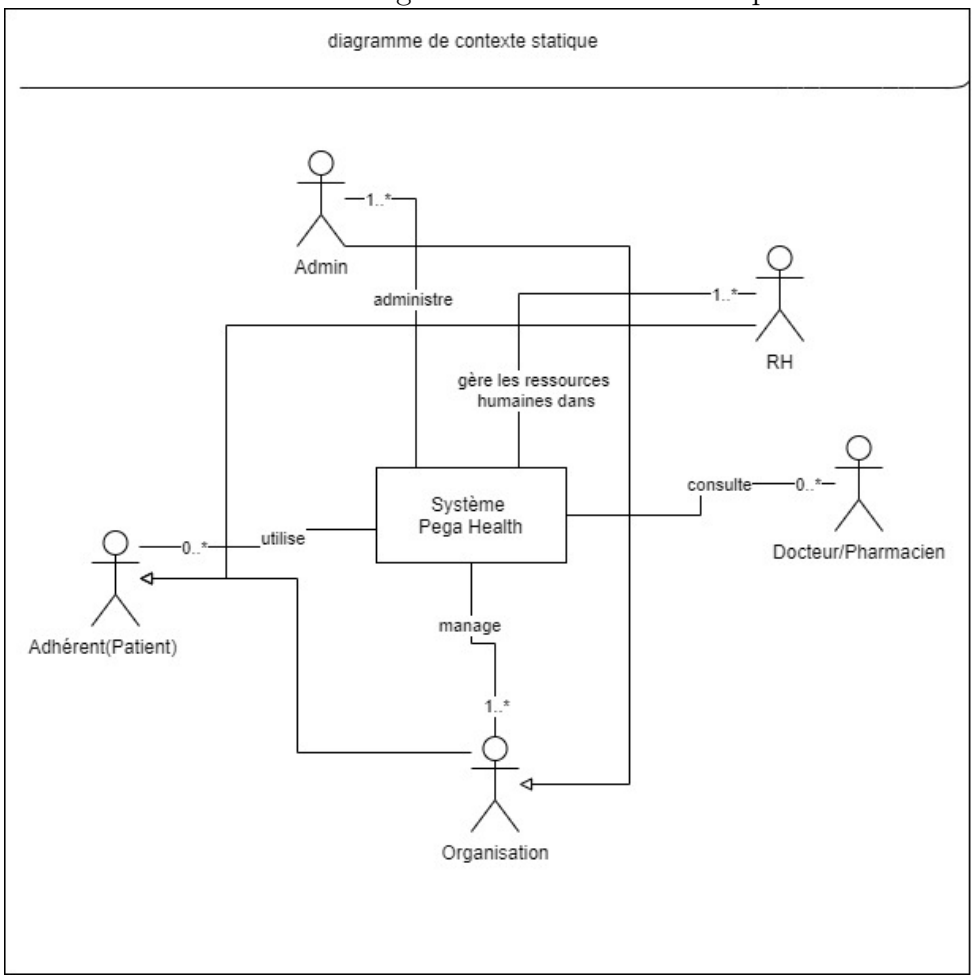
FIGURE 1 – Diagramme d'acteurs.



2.1.3.2 Diagramme de contexte statique

Le diagramme de contexte suivant identifie l'environnement extérieur au système d'assurance santé Pega Health :

FIGURE 2 – Diagramme de contexte statique.



2.1.3.3 Intentions d'acteurs

A partir du problème posé, nous allons lister les fonctions qui pourront être réalisées durant ce projet :

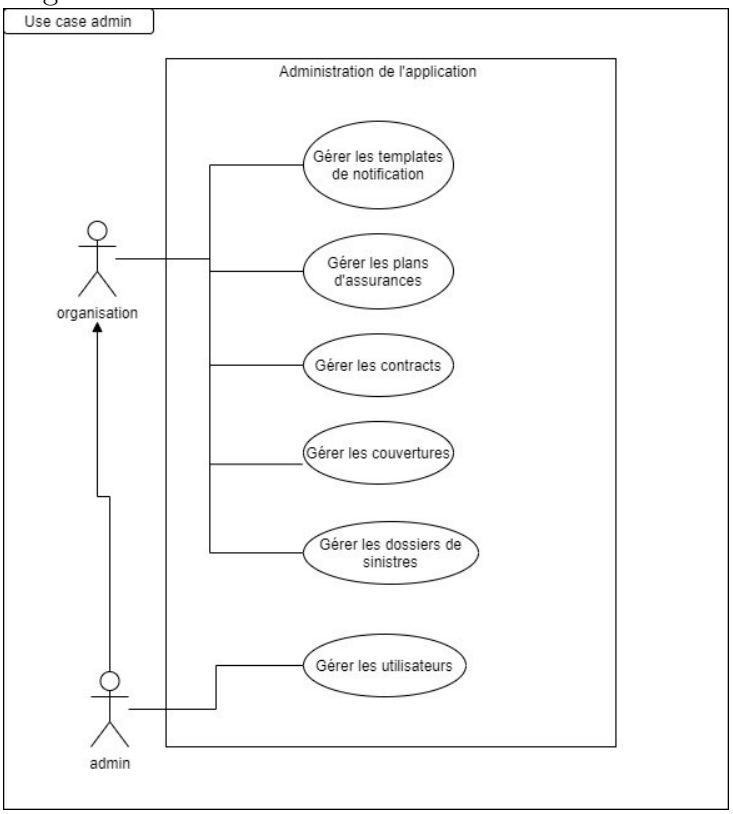
Référence	**Fonction**	**Intention d'acteur**	**Acteurs**
C1	Ajouter un docteur, patient, organisation	**Gérer les utilisateurs**	Admin
C2	Ajouter un plan d'assurance et préciser son prix et ses garanties	**Gérer les plans d'assurance**	— Admin — Organisation
C3	— Inviter un nouveau salarié(patient) à un plan d'assurance — Lui rappeler de souscrire — Le retirer pour résilier son contrat	**Gérer les contrats**	— Admin — Organisation — RH
C4	Demander un remboursement	**Demander des remboursements**	Patient/Adhérent
C5	Lister les demandes de remboursements	**Gérer les dossiers de sinistres**	— Admin — Organisation
C6	Choisir un plan d'assurance	**souscrire à un plan d'assurance**	Patient/Adhérent
C7	Pour chaque type de notification, personnaliser le message à envoyer	**Gérer les templates de notification**	— Admin — Organisation
C8	Renseigner pour chaque patient un dossier médical	**Ecrire des visites médicales**	Docteur

TABLE 2.1 – Intentions d'acteurs

2.1.3.4 Identification des cas d'utilisation

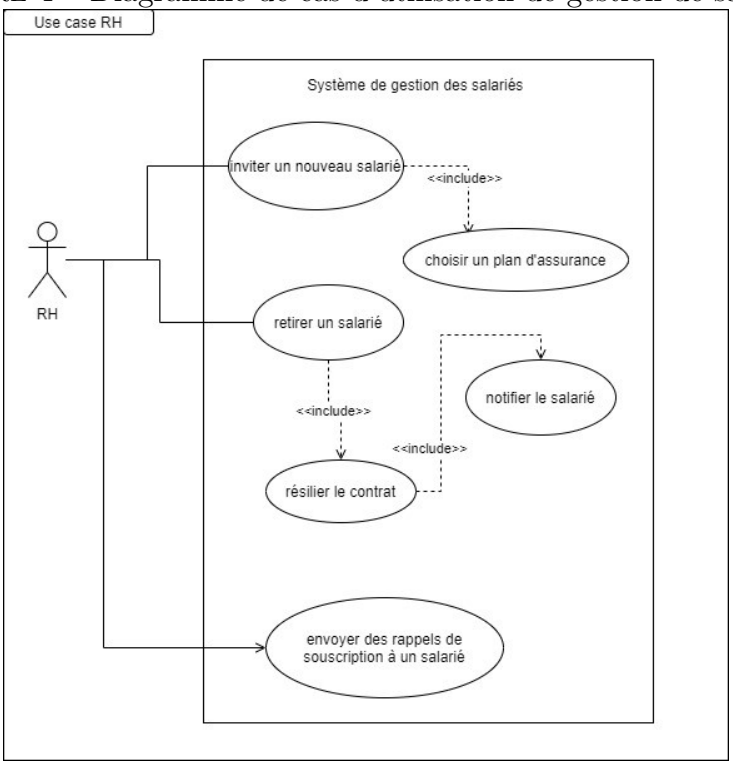
Afin de représenter les fonctionnalités du système, nous avons élaboré des diagrammes de cas d'utilisation pour chaque acteur :

FIGURE 3 – Diagramme de cas d'utilisation de l'administration de l'application



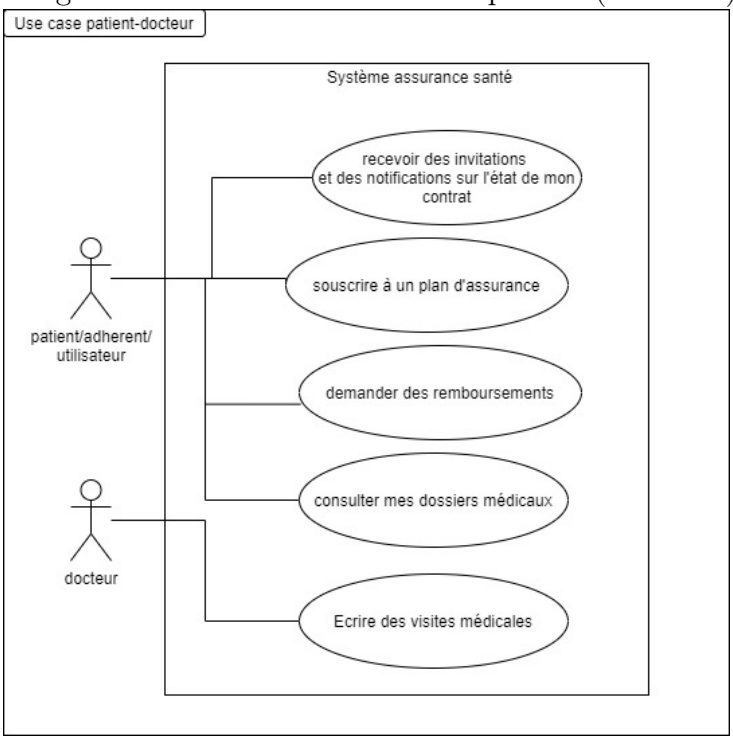
Le diagramme au-dessus montre la hiérarchie entre l'admin et l'organisation, une organisation ne peut pas gérer les utilisateurs de l'application.

FIGURE 4 – Diagramme de cas d'utilisation de gestion de salariés



Le rôle d'un RH est de manipuler les contrats des salariés, et n'a droit de contrôler que les adhérents appartenant à son organisation.

FIGURE 5 – Diagramme de cas d'utilisation du patient (adhérent) et médecin



Un patient est un salarié ou un simple utilisateur qui bénéficie des services santé, a droit de choisir un plan d'assurance convenable à ses attentes à travers des contrats gérés par une organisation d'assurance.

2.1.4 Besoins techniques

2.1.4.1 Environnement de développement

La préparation d'un environnement de développement est indispensable pour le bon départ du développement de notre application. Pour cela, une configuration matérielle et logicielle bien définie était impérative. Afin de justifier notre choix, Voici les technologies utilisées :

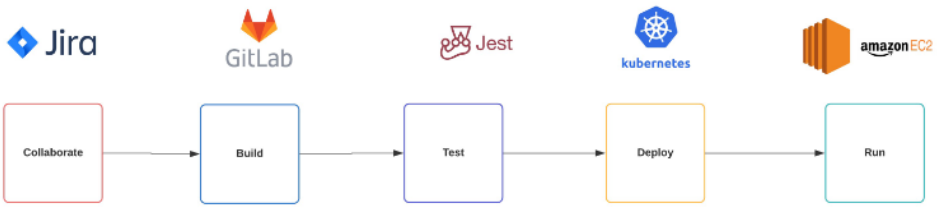
— React Native [2]
— ReactJs [3]
— NodeJs [4]
— GraphQL [5]
— Architecture microservices [6]
— Kafka [7]
— PostgreSQL [8]

— Docker [9]
— Kubernetes [10]
— Typescript [11]

2.1.4.2 Démarche DevOps

La démarche DevOps a pour objectif une gestion intégrale des processus d'ingénierie informatique et rapprochement entre les développeurs et les opérateurs.
Parmi les meilleurs processus qu'une équipe DevOps puisse mettre en place est le **(CI/CD)**.
L'approche **(CI/CD)** garantit une automatisation et une surveillance continues tout au long du cycle de vie des applications, des phases d'intégration et de test jusqu'à la distribution et au déploiement. Ensemble, ces pratiques sont souvent désignées par l'expression « pipeline CI/CD » et elles reposent sur une collaboration agile entre les équipes de développement et d'exploitation.
Le diagramme suivant montre la pipeline **(CI/CD)** de la future application :

FIGURE 6 – Pipleline CI/CD Pega Health.

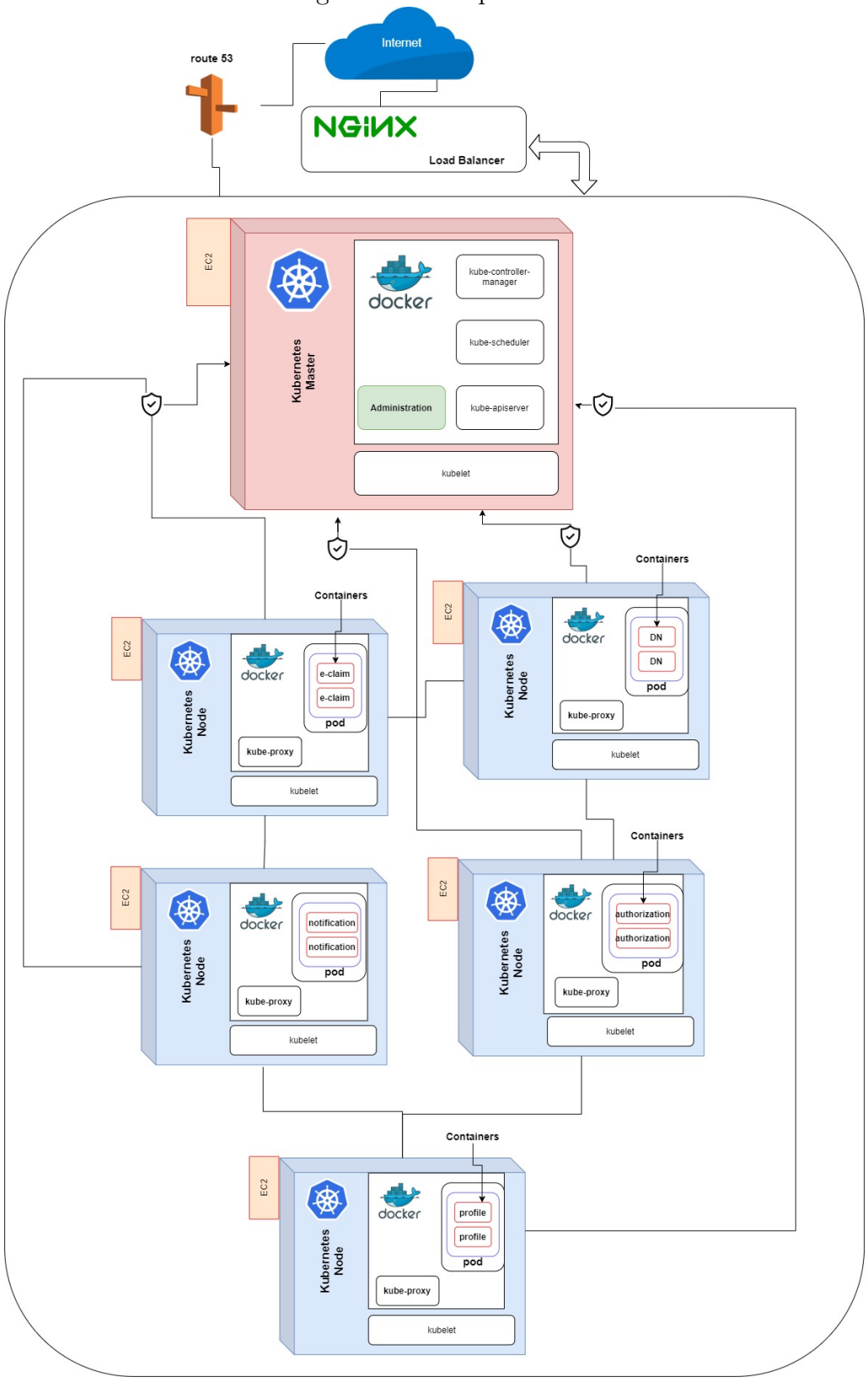


2.1.5 Modélisation des besoins techniques

2.1.5.1 Diagramme de déploiement des microservices

L'application sera décomposée en plusieurs microservices :
— **E-claim** pour la gestion des remboursements et plans d'assurance ;
— **Profile** pour la gestion des différents types de profils : admin, RH, médecin, patient(adhérent) . . . ;
— **Authorization** pour filtrer les accès aux données selon le profil connecté ;
— **Notification** pour l'envoi des emails et SMS ;
— **DN** pour l'usage du pattern dénormalisation qui réunit plusieurs sources d'entrée normalisées en une structure de données lisible qu'un client peut consommer.
Le diagramme suivant montre l'architecture de déploiement de ces microservices :

FIGURE 7 – Diagramme de déploiement des microservices



Chaque **kubernetes Node** est un **WN** (une machine physique ou une machine virtuelle) qui détiendra toutes les ressources nécessaires afin de garantir l'exécution d'un

ou plusieurs pods. Cette entité va héberger tous les services dont nous aurons besoin.

Chaque WN contiendra un microservice avec ses propres ressources y compris sa base de données. Ces WN vont être administrées par **Kubernetes Master**.

Pour répartir uniformément les charges de travail sur plusieurs serveurs nous allons utiliser **Nginx** comme **load balancer**.

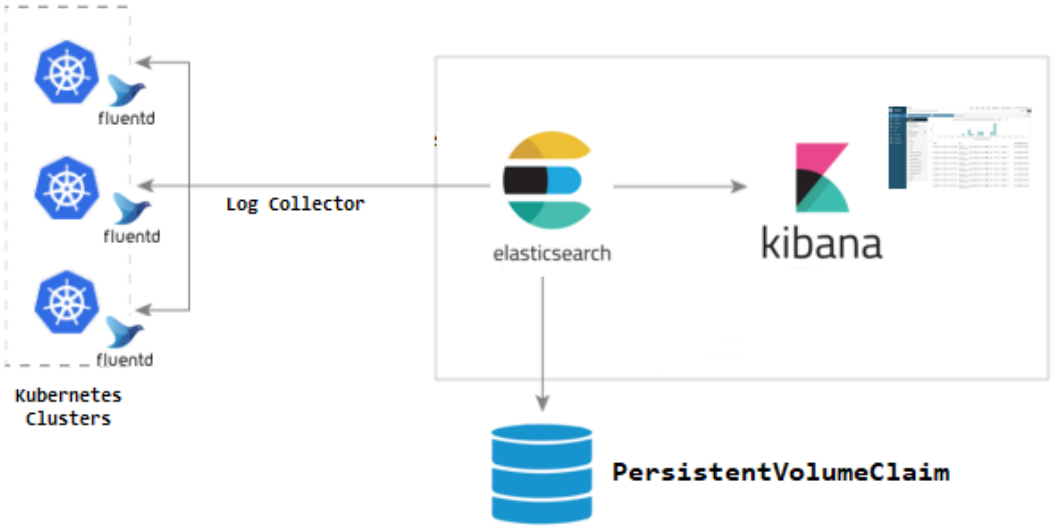
Pour acheminer les utilisateurs finaux vers la future application, Les requêtes des utilisateurs seront connectées à l'infrastructure à travers **Amazon route 53** s'exécutant dans AWS. Ce service permet également d'acheminer les utilisateurs vers une infrastructure extérieure à AWS. **Amazon route 53** est un service Web de DNS dans le cloud hautement disponible et évolutif.

2.1.5.2 Diagramme de déploiement de logs

Pour détecter les demandes suspectes dans la future application, enregistrer toutes les tentatives de connexion et contrôler les fraudes, nous aurons besoin de mettre en place une solution de gestion de logs pour enregistrer toutes les activités, ainsi pour des raisons de débogage.

Le diagramme suivant montre l'architecture de centralisation de logs :

FIGURE 8 – Diagramme de déploiement de logs



Nous collecterons les logs de serveur à travers **Fluentd** [12], les enverrons vers un cluster **ElasticSearch** [13], et nous configurerons **Kibana** [14] pour visualiser ces données.

2.2 Méthodologie de gestion de projet

2.2.1 SCRUM

Aujourd'hui, l'aptitude d'une entreprise à être plus agile [15] dépend de plus en plus de sa capacité à bâtir et à proposer une expérience client de qualité. Si les nouvelles technologies offrent un appui précieux, la réussite repose sur l'existence d'une vision et d'une stratégie cohérente, à tous les niveaux de l'entreprise.

L'environnement de l'entreprise d'accueil **Taillis Labs** [1] est un environnement agile [15] qui adopte SCRUM [16] dans sa gestion de projets.

Nous avons adopté **Jira** [17] comme outil d'organisation des tâches, et la durée de chaque sprint est de 4 semaines.

2.2.2 Kanban

Les tâches de projet seront réparties dans un tableau Kanban [18] et divisées en quatre parties :

- À faire ;
- En cours de développement ;
- À tester ;
- Réalisé.

2.3 Conclusion :

Lors de cette phase nous avons essayé d'exprimer le fonctionnement de notre système en se basant principalement sur le diagramme de contexte statique, diagramme de cas d'utilisation et diagramme de déploiement. Après avoir décrit les besoins non fonctionnels, fonctionnels et techniques attendus de notre application qui consistent à mettre en place une démarche de développement, nous pouvons ainsi entamer la prochaine étape de l'analyse et conception.

Chapitre 3

Sprints

Introduction :

Ce chapitre fait l'objet d'une présentation des différentes tâches réalisées durant chaque sprint afin de mettre en œuvre une version stable en mode production de notre système à travers la phase d'analyse, conception, réalisation (développement) et test.

3.1 Gestion de profils

Un profil correspond à un ensemble de droits qui définissent les actions que peut effectuer un utilisateur dans l'application. [19]

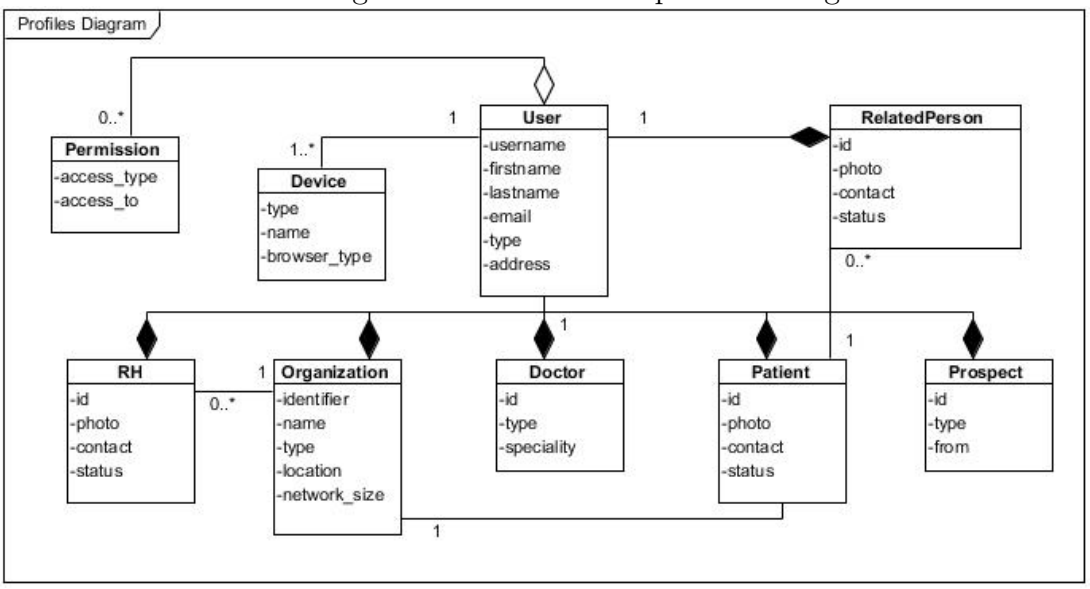
3.1.1 Analyse

Plusieurs utilisateurs demandent des traitements sur les ressources de Pega Health, toutefois ces utilisateurs n'ont pas les mêmes besoins : certains peuvent nécessiter de modifier des données, tandis que les autres ne l'utiliserons que pour la consultation et l'affichage.
Il est possible de définir des permissions pour chaque utilisateur : créer un profil.
Un micro-service doit exister pour autoriser ou bloquer les besoins de chaque utilisateur selon son profil.

3.1.2 Conception : diagramme de classe

Pour distinguer les types d'utilisateurs, chaque profil doit avoir une classe défini :
la classe **RelatedPerson** modélise une relation personnelle ou professionnelle avec le patient exemple de frère, soeur, parent, chaque relation est liée à un seul **patient** qui appartient à une **organisation** gérée par un profil **RH**.
Un **Prospect** caractérise un client potentiel.
Chaque profil se compose d'un **User** qui contient les données d'authentification et **Device** qui enregistre la liste des périphériques à travers lesquels il se connecte.
Pour limiter l'accès à certaines ressources, nous avons configuré une classe dédiée aux **Permissions**.

FIGURE 9 – Diagramme de classe des profils de Pega Health



3.1.3 Tests fonctionnels

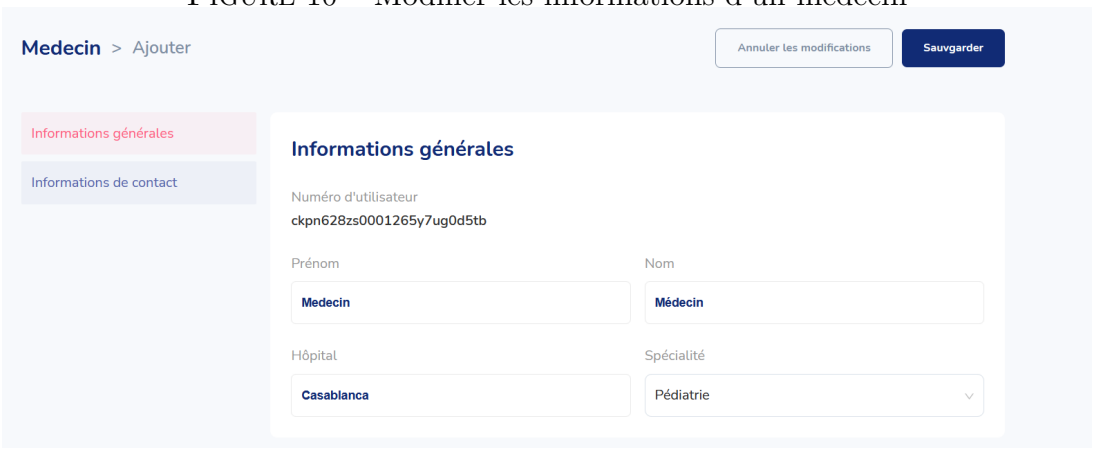
Cas de test	Comportement attendu	Résultat
L'administrateur peut ajouter et/ou modifier un utilisateur	Les données sont conformes et bien enregistrées	CONFORME
Afficher les patients, les médecins et les organisations séparément	Les profils de même type s'affiche dans la même section	CONFORME
Un patient demande de lister les patients de l'application	Le serveur affiche un message d'erreur	CONFORME

TABLE 3.1 – Tests fonctionnels de sprint gestion de profils

3.1.4 Réalisation

La figure suivante montre une capture d'une des fonctionnalités réalisées durant ce sprint :

FIGURE 10 – Modifier les informations d'un médecin



La gestion des utilisateurs est une étape très intéressante dans une application à accès multiples, ce qui nous a amené à travers ce sprint à définir un pattern précis pour le management des rôles, puis nous avons validé cette solution par le test.

3.2 Gestion de plans d'assurance

L'assurance santé est un type d'assurance qui offre à l'assuré une couverture des frais médicaux en cas d'urgence sanitaire.
Le plan d'assurance santé choisi par l'assuré offre une couverture pour différentes dépenses, notamment les frais chirurgicaux, les maladies graves et des frais de soins, etc.

Un contrat est une obligation entre l'assureur et l'assuré dans lequel la compagnie d'assurance fournit une couverture financière pour les frais médicaux encourus par l'assuré et prévoit de le rembourser selon le type de plan d'assurance offert.

3.2.1 Analyse

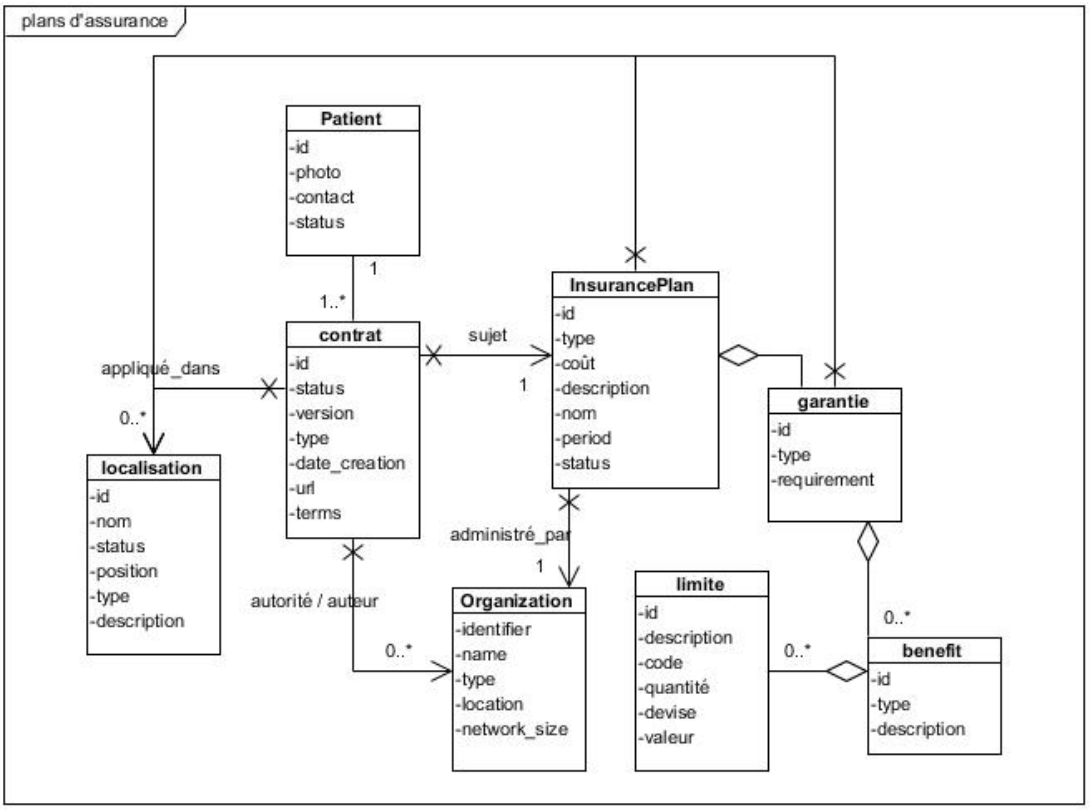
Pega Health met à disposition un service complet et facile à gérer permettant aux entreprises de souscrire à une assurance santé et d'y inscrire leurs salariés en quelques clics.
En effet,les dirigeants pourront réaliser l'ensemble des démarches depuis la plateforme.
Un RH peut inviter tous ses salariés à adhérer avec un simple mail, ils s'inscrivent seuls et sont couverts immédiatement (familles incluses).

3.2.2 Conception : diagramme de classe

Une compagnie d'assurance est une organisation qui offre des **plans d'assurance santé** de plusieurs types : vision, mental, dentaire . . . , chaque plan **garantie** de couvrir des frais médicaux, pharmaceutiques et d'hospitalisation occasionnés par une maladie, un accident une maternité ou autre, un plan assurance santé contient des plafonds et des **limites territoriales** liées au type de garantie.

L'achat d'une offre dépend d'un **contrat** instancié par les deux parties : patient et organisation.

FIGURE 11 – Diagramme de classe de plans d'assurance



3.2.3 Tests fonctionnels

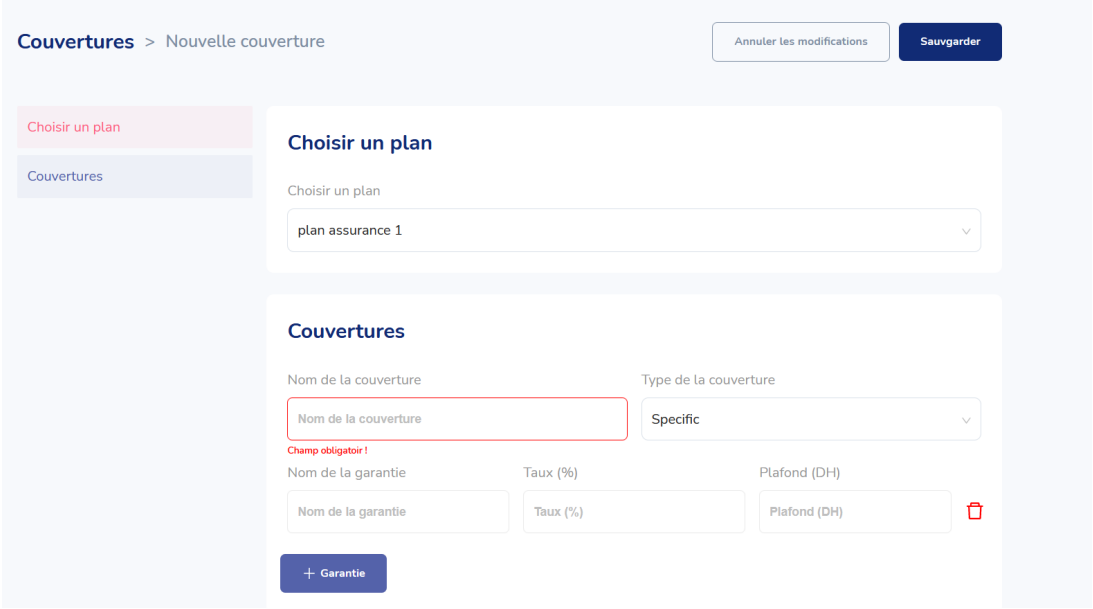
Cas de test	**Comportement attendu**	**Résultat**
L'administrateur peut ajouter et/ou modifier un plan d'assurance	Les données sont conformes et bien enregistrées	CONFORME
L'organisation peut ajouter des couvertures en spécifiant leurs plafonds et types	Les données sont conformes et bien enregistrées	CONFORME
Un patient reçoit un URL pour débuter le processus d'inscription à une offre	L'email s'envoie correctement avec un URL valide	CONFORME

TABLE 3.2 – Tests fonctionnels de sprint gestion de plans d'assurance

3.2.4 Réalisation

La figure suivante montre une capture d'une des fonctionnalités réalisées durant ce sprint :

FIGURE 12 – Ajout des couvertures



Nous avons détaillé les informations qui constituent un plan d'assurance, puis nous avons analysé le sujet pour préciser les spécifications d'un contrat d'assurance santé, puis entamé la phase de conception, et finalement nous avons pu mettre en valeur les résultats attendus après avoir implémenté la solution trouvée.

3.3 Gestion des remboursements

En souscrivant à une mutuelle santé, vous bénéficiez d'un remboursement des frais des soins complémentaire à celui de l'assurance santé. Ce complément peut représenter une partie ou la totalité du montant dépensé, c'est-à-dire de la partie non prise en charge par l'assurance santé.

La hauteur de remboursement par la mutuelle dépend des garanties souscrites, elles-mêmes associées aux diverses catégories de soins. Le régime d'assurance santé pose, pour chaque type de dépense, une base de remboursement (BR) liée au type de plan d'assurance santé.

Les modalités de demande de remboursement des frais de santé garantis par la mutuelle dépendent du contrat souscrit. Dans certains cas, l'assuré n'aura rien à envoyer : le remboursement sera automatique et rapide. Dans d'autres, il faudra faire parvenir à la mutuelle une lettre de demande de prise en charge en joignant certains documents justificatifs - le cas de Pega Health -.

Les documents justificatifs lors d'une demande de remboursement devront détailler précisément les soins reçus et la somme payée.
Il pourra notamment s'agir :

- d'une feuille de soins ;
- d'une ordonnance ;
- d'une facture de paiement détaillée et acquittée.

Par exemple, en cas d'hospitalisation, vous devrez transmettre un bulletin d'hospitalisation et une facture réglée. Pour vos lunettes, il faudra envoyer une facture des sommes acquittées ou un devis. [20]

3.3.1 Analyse

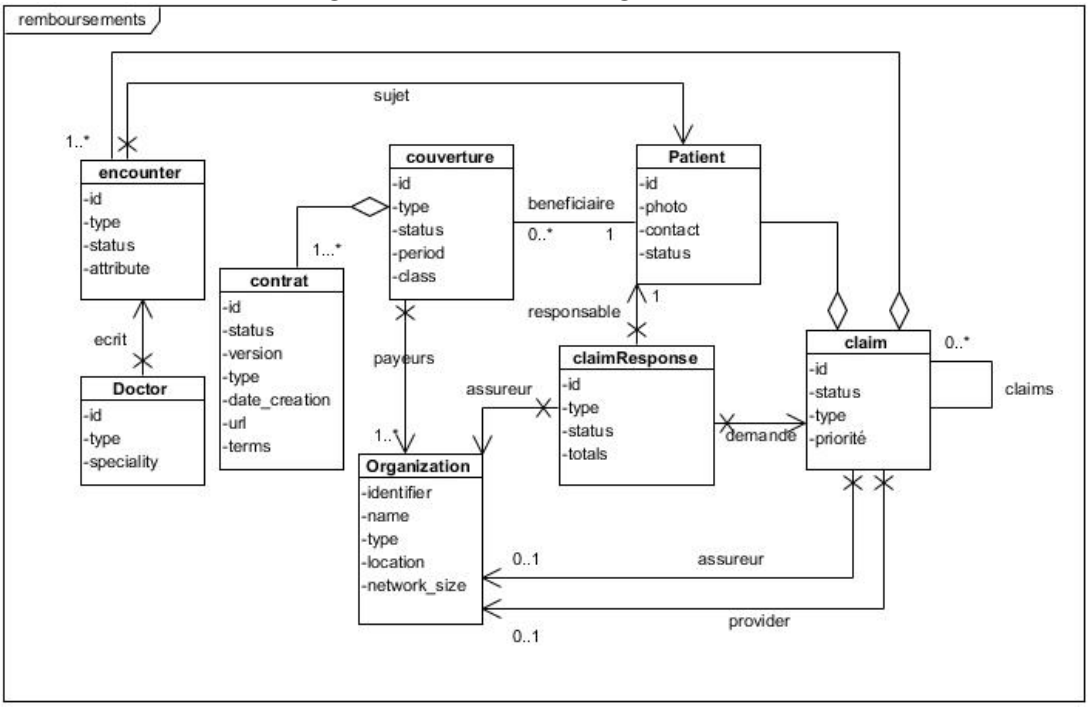
Les demandes de remboursement sont envoyées à l'assureur depuis l'espace personnel du patient assuré.
Il suffit de scanner les factures et de les envoyer depuis son smartphone.

3.3.2 Conception : diagramme de classe

Lorsqu'un patient rencontre un médecin (**Doctor**) pour une consultation médicale, ce dernier établit un diagnostic enregistré comme "**encounter**" dont le sujet est le **patient**. Lorsque le patient assuré envoie une demande de remboursement "**claim**" sur ces soins à l'organisation d'assurance, il sélectionne les documents (encounter) qui légitiment sa

demande. La réponse à cette demande est un "**claimResponse**" qui vérifie le type de couverture de l'assuré et les termes de son contrat.

FIGURE 13 – Diagramme de classe de gestion des remboursements



3.3.3 Tests fonctionnels

Cas de test	**Comportement attendu**	**Résultat**
Un médecin ajoute un dossier médical pour son patient	Les données sont conformes et bien enregistrées	CONFORME
Un médecin est capable de lister ses patients et voir leurs consultations	Les patients listés ont déjà eu un diagnostic chez ce médecin	CONFORME
Un patient demande un remboursement	La demande est bien enregistré et envoyé pour le traitement	CONFORME

TABLE 3.3 – Tests fonctionnels de sprint gestion des remboursements

3.3.4 Réalisation

Les figure suivantes montrent quelques captures des fonctionnalités réalisées durant ce sprint :

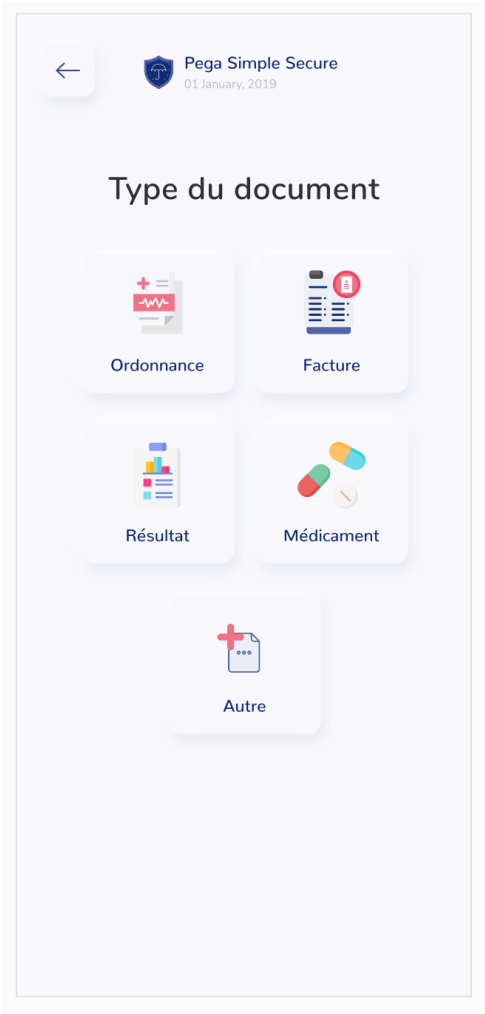


FIGURE 14 – Selectionner le type du document

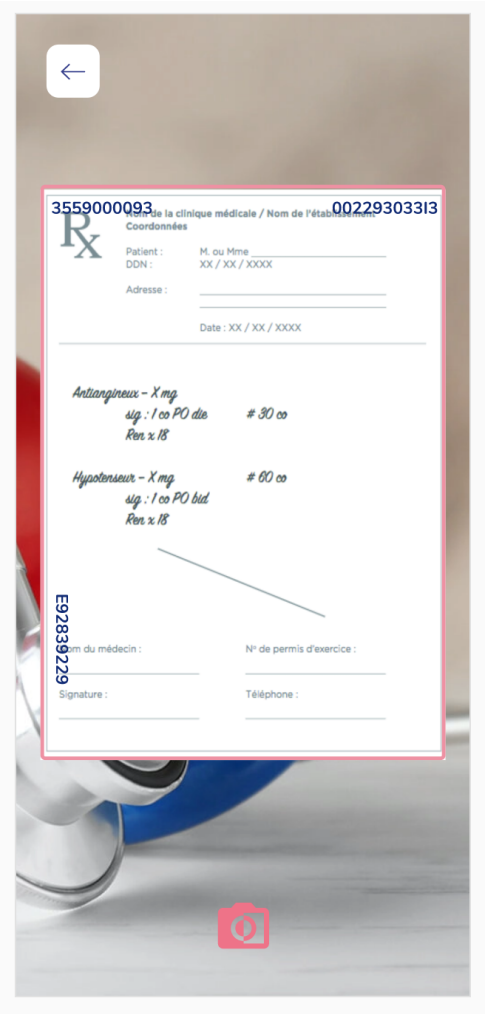


FIGURE 15 – Scanner des pièces justificatives à attacher à la demande

Sprints

Conclusion :

Nous avons essayé d'exprimer sur chaque sprint les fonctionnalités abordées avec des diagrammes de classe.
A la fin de chaque release/sprint clôturé en un mois, nous étions capables de livrer des résultats compatibles approuvés par les tests.
L'application est toujours en cours de développement et l'ajout de nouvelles fonctionnalités est maintenue sur les prochains sprints.

Conclusion

Dernièrement, toutes les organisations IT essaient de mettre en place une architecture micro-service - le cas de Taillis Labs aussi - à travers ce stage, j'ai appris que cette architecture constitue **un défi en matière débogage :** chaque microservice faudra être testé individuellement, et ensuite tester tout l'ensemble, **un défi en matière de dépendance :** chaque changement dans un microservice impacte la version des autres, **un défi en matière de mise à jours des bases de données cloisonnées ou sur les migrations**, et un autre **défi lié au domaine :** l'apparition d'un nouveau besoin conceptuel ou d'une nouvelle entité (table dans la base de données) nous incite à mener une discussion lente afin de trouver sur quel microservice faudra l'ajouter, cette discussion impacte la vélocité de l'équipe et par conséquent le livrable de sprint.

SCRUM quant à elle n'est pas facile à mettre en place si l'équipe n'est pas expérimentée, le nombre de user stories à traiter sera faible par rapport aux exigences du client et par conséquent des retards de livraison, des autres problèmes liés à la documentation considéré comme limite des méthodes agiles : **documentation moins élaborée**, étant donné que les développements débutent rapidement, les exigences du produits sont clarifiées juste à temps pour le développement, lorsque de nouveaux membres rejoignent l'équipe, ils ne connaissent pas les détails de certaines fonctionnalités ni la manière dont elles doivent fonctionner ce qui peut occasionner des difficultés et une **adaptation plus longue**.

Toutefois, durant ce stage j'ai appris de bonnes pratiques de développement, plus d'exigence conceptuel orienté métier à travers une architecture micro-service pilotée par le domaine qui m'a rendue consciente du métier assurance santé.

Bibliographie

[1] Taillis Labs. Site officiel : https://taillislabs.com/.

[2] React Native. Site officiel : https://reactnative.dev/.

[3] React js. Site officiel : https://fr.reactjs.org/.

[4] NodeJs. Site officiel : https://nodejs.org/en/.

[5] GraphQL. Site officiel : https://graphql.org/.

[6] Architecture microservice. Article de Taillis Labs sur medium : https://medium.com/@taillisartists/saga-eab68b6ddc31.

[7] Kafka. Site officiel : https://kafka.apache.org/.

[8] Postgresql. Site officiel : https://www.postgresql.org/.

[9] Docker. Article sur le choix de docker : https://www.infoworld.com/article/3310941/why-you-should-use-docker-and-containers.html.

[10] Kubernetes. Site officiel : https://kubernetes.io/fr/.

[11] Typescript. Site officiel : https://www.typescriptlang.org/.

[12] Fluentd. Site officiel : https://www.fluentd.org/.

[13] ElasticSearch. Site officiel : https://www.elastic.co/elasticsearch/.

[14] Kibana. Site officiel : https://www.elastic.co/kibana.

[15] Nutcache. https://www.nutcache.com/fr/blog/agilite-entreprise-entreprise-agile/.

[16] David Galiana. https://www.planzone.fr/blog/quest-ce-que-la-methodologie-scrum.

[17] Atlassian. Site officiel : https://www.atlassian.com/fr/software/jira.

[18] journaldunet. https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443832-kanban-une-methode-agile-de-gestion-de-projet-visuelle-et-continue/.

[19] openflyers. Site : https://openflyers.com/fr/doc/of4/Gestion-des-profils.

[20] Antoine Fruchard, 2020. https://reassurez-moi.fr/guide/mutuelle-sante/lettre-demande-remboursement.